FedLAD: A Linear Algebra Based Data Poisoning Defence for Federated Learning

Qi Xiong, Hai Dong, Nasrin Sohrabi and Zahir Tari

FedLAD: A Linear Algebra Based Data Poisoning Defence for Federated Learning

Anonymous

Abstract. Sybil attacks pose a significant threat to federated learning, as malicious nodes can collaborate and gain a majority, thereby overwhelming the system. Therefore, it is essential to develop countermeasures that ensure the security of federated learning environments. We present a novel defence method against targeted data poisoning, which is one of the types of Sybil attacks, called Linear Algebra-based Detection (FedLAD). Unlike existing approaches, such as clustering and robust training, which struggle in situations where malicious nodes dominate, FedLAD models the federated learning aggregation process as a linear problem, transforming it into a linear algebra optimisation challenge. This method identifies potential attacks by extracting the independent linear combinations from the original linear combinations, effectively filtering out redundant and malicious elements. Extensive experimental evaluations demonstrate the effectiveness of FedLAD compared to five well-established defence methods: Sherpa, CONTRA, Median, Trimmed Mean, and Krum. Using tasks from both image classification and natural language processing, our experiments confirm that FedLAD is robust and not dependent on specific application settings. The results indicate that FedLAD effectively protects federated learning systems across a broad spectrum of malicious node ratios. Compared to baseline defence methods, FedLAD maintains a low attack success rate for malicious nodes when their ratio ranges from 0.2 to 0.8. Additionally, it preserves high model accuracy when the malicious node ratio is between 0.2 and 0.5. These findings underscore FedLAD's potential to enhance both the reliability and performance of federated learning systems in the face of data poisoning attacks. For further details, we have open-sourced our work at https://gitlab.com/Anonymous.

1 Introduction

Federated learning (FL), proposed by [17], is a machine learning (ML) network with a star topology in which computing nodes submit their locally trained ML models to a central server to obtain a global ML model using an average merging method (referred to as FedAVG hereinafter). The global ML model is then sent back to the nodes to continue their ML tasks. This approach is designed to protect the data privacy of nodes since it allows ML models to be trained without data being collected on a central server. However, this architecture is vulnerable to various attacks in which malicious nodes can attack the global model. Especially in Sybil attacks, malicious nodes can collude with each other and can easily occupy the majority, rendering FL systems unreliable [10, 21]. In Sybil attacks, various fundamental attacks can be performed. According to [1], these attacks

can be classified into model poisoning attacks, such as backdoor attacks [3, 4], and data poisoning attacks, such as label flipping [19]. These attacks can be further classified into targeted attacks and untargeted attacks [20, 18]. In untargeted attacks, malicious nodes aim purely to lower the performance of the global ML model [20]. In targeted attacks, malicious nodes have specific goals for attacking the FL system [20]. Backdoor attacks [3, 4], which insert triggers into the ML model [12] to improve the performance of malicious labels, are a common example of such targeted attacks [19, 12]. As highlighted in [19], targeted label flipping attacks are particularly effective against FL systems and can be executed even by adversaries with limited capabilities. Hence, our work focuses on defending against **targeted** data poisoning attacks.

Existing defence methods that focus on data poisoning defence in FL can be classified into the categories of *clustering-alike defence*, such as Sherpa [18], which focuses on applying clustering techniques such as HDBSCAN to identify malicious clusters base on their ML models' interpretability; and *robust training-based defence*, such as the median merging method (i.e., aggregation) [24, 22], which replaces the average merging method with an attack-aware merging method. All the existing methods have the limitation of only tolerating a certain proportion of malicious nodes. For example, in clustering-alike defence methods, the maximum tolerance of malicious nodes is 50%. If the ratio of malicious nodes exceeds 50%, the major cluster is dominated by malicious nodes, making it unreliable as an indicator of whether the cluster is benign or malicious. This limitation makes federated learning vulnerable to Sybil attacks in which malicious nodes can collude with each other and occupy the majority [10, 21]. Robust training-based defence methods consider the impact of malicious ML models during the training or aggregation process. However, when the majority of ML models are malicious, these methods can be overwhelmed.

To address these limitations of existing methods, this paper proposes a Linear Algebra-based Defence (FedLAD) to defend against data poisoning attacks in FL. FedLAD models the aggregation process in FL as a linear combination. By finding the independent linear combination from the original linear combination, we can filter out redundant and malicious ML models, hence defend against malicious attacks such as data poisoning attacks. Since the malicious ratio has a low impact on the independent linear combination, FedLAD has a high tolerance for malicious nodes. Furthermore, to improve computational efficiency, we introduce a parallel optimisation algorithm based on sub-matrix splitting, which allows FedLAD to leverage parallel computing. This represents a significant advancement over existing defence approaches, whose reliance on sequential algorithms,

such as clustering, limits their parallelizability.

The primary contributions of this paper are as follows:

- We propose a defence method against targeted data poisoning attacks in Federated Learning (FL), named FedLAD, which is based on linear algebra techniques. This method demonstrates a high tolerance for malicious nodes; for instance, our experiments show that FedLAD can remain effective even when 70% of the nodes are malicious in some datasets such as AG_NEWS.
- To the best of our knowledge, we are the first to leverage parallelisation to accelerate the process of finding the independent linear combination from its original linear combination. Moreover, we provide a formal mathematical proof demonstrating that this linear algebra problem can be effectively solved in a parallel computing environment. In contrast to FedLAD, existing defence methods are unable to exploit parallel architectures, as their underlying algorithms, such as clustering, are inherently sequential and based on linear logic.

We conducted experiments with malicious ratios ranging from 0.2 to 0.8 using three datasets: CIFAR10, CIFAR100, and AG_NEWS. These experiments are performed with non-IID distributions. The results demonstrate that FedLAD is more robust to attacks compared to existing methods, including Sherpa [18], CONTRA [1], Median [24], Trimmed Mean [23], and Krum [5]. The baseline methods show a significant drop in model accuracy when the malicious ratio exceeds 50%. In contrast, FedLAD consistently maintains high model accuracy when the malicious ratios surpass 50%, effectively mitigating the impact of malicious nodes where other methods fail. We also conducted experiments to compare the running times of FedLAD with serial and parallel versions. Our findings indicate that with the parallel optimisation we proposed, FedLAD runs three to four times faster than its serial version. This demonstrates the advantages of our parallel optimisation approach.

The rest of the paper is organised as follows: Section 2 introduces the background knowledge of linear algebra that is relevant to our proposed method. Section 3 provides solution details of FedLAD. Section 4 discusses the various experimental results. Section 5 summarises some of the existing works. We conclude the paper in Section 6.

2   Background

Our proposed defence method is based on theories from linear algebra. Therefore, we list all related definitions and theories in this section according to the book [2].

Definition 1. *A **linear combination** is a set of vectors in a vector space [2, p. 46]. The equation below shows an example of a linear combination:*

$$a_1 v_1 + a_2 v_2 + ... + a_n v_n, \qquad (1)$$

where a_i is a coefficient, v_i is a vector.

Definition 2. *A **span** is all the linear combinations of the vectors that form a subspace in a vector space [2, p. 47]. The equation below shows an example of a span:*

$$span(v_1, ..., v_n) = a_1 v_1 + a_2 v_2 + ... + a_n v_n, \qquad (2)$$

where $a_1 v_1 + a_2 v_2 + ... + a_n v_n$ is the set of linear combinations that spans the vectors $v_1, ..., v_n$.

Definition 3. *A linear combination is called an **independent linear combination** if the only way to make the combination equal zero is by setting all coefficients to zero [2, p. 50]. It means that no vector in the set can be represented by a linear combination of other vectors in the same set. The equation below shows an example of an independent linear combination:*

$$a_1 v_1 + a_2 v_2 + ... + a_n v_n = 0 \iff a_1 = a_2 = ... = a_n = 0, \quad (3)$$

where v_i is a vector from the linear combination in the equation.

Definition 4. *A **basis** of a vector space is a set of linear independent vectors that span that vector space [2, p. 57].*

Definition 5. *A **rank** of a matrix is the number of linearly independent vectors in the rows or columns of that matrix [2, p. 77].*

Definition 6. *The vectors from an independent linear combination are defined as **independent vectors**. The vectors that are represented by the independent linear combination are defined as **dependent vectors**.*

Theorem 1. *Every vector space that has a finite dimension has a basis [2, p. 59].*

Theorem 2. *The number of vectors in independent linear combinations of a vector space is unique (i.e., all bases in a vector space have the same dimension) [2, p. 62]. Accordingly, the rank of a matrix is unique since a matrix can be viewed as a list of vectors.*

Theorem 3. *The row rank and column rank of a matrix are the same [2, p. 78].*

3   The Proposed Method

3.1   Threat Model

Malicious nodes engage in targeted label flipping on their local datasets, which are used to train their local machine learning models. Because these malicious models are influenced by corrupted datasets, we propose a method to detect the compromised ML models on the server side as a defence against data poisoning attacks. In light of this approach and our emphasis on mitigating targeted data poisoning attacks, we model the attacks from these malicious nodes with the following equation:

$$\arg \min_{w_{m,t}} F(w_{m,t}, x_{m,t}, y_{m,t}), \qquad (4)$$

which is to train the local ML model with polluted data $x_{m,t}$ and label $y_{m,t}$ by optimising the loss function F. The optimised local poisoned ML models $w_{m,t}$ will be merged into the global ML model if not detected.

3.2   Problem Definition

In FL, every node trains its local ML model, and those local ML models are merged into a global model. The process can be formulated as:

$$w_{g,t} = FedAvg(S_t) = \frac{1}{|S_t|} \sum_i w_{i,t}, \qquad (5)$$

where $w_{g,t}$ is the global ML model parameter at the round t, $w_{i,t}$ is the i-th local ML model at the round t. In our case, there are malicious nodes in the system. We define the set M to indicate a malicious node set. We define the set B to indicate a benign node set. We

define S_t to indicate the whole set at the round t that contains M_t and B_t (i.e., $S_t = M_t \cup B_t$). The problem of our work is to optimise the equation below:

$$w_{b,t} - FedAvg(B_t) \to 0, \qquad (6)$$

where $w_{b,t}$ is the model parameter merged by nodes classified as benign. The above equation means that $w_{b,t}$ should be close to the global ML model with malicious ML models excluded so that their difference can approach 0. We get $w_{b,t}$ by the below equation:

$$w_{b,t} = \frac{1}{\sum_i^{|S_t|} f^d(w_{i,t})} \sum_i^{|S_t|} w_{i,t} \times f^d(w_{i,t}), \qquad (7)$$

where f^d is the malicious node detection function that classifies a node as either benign or malicious. The output is 0 if the node belongs to the malicious set; otherwise, the output is 1. This is represented by the following equation:

$$f^d(w_{i,t}) = \begin{cases} 1, w_{i,t} \in B_t \\ 0, w_{i,t} \in M_t \end{cases} \qquad (8)$$

3.3 Solution Details

3.3.1 Preliminary

We model the aggregation process (i.e., merge local ML models into a global ML model) in FL as a linear combination in which the global ML model is represented by a linear combination of local ML models. The Global and Local ML models form a vector space. The equation below illustrates this concept:

$$a_1 w_1 + a_2 w_2 + ... + a_n w_n = w_g \ s.t. \{\sum_{i=1}^{n} a_i = 1, a_i \ge 0\}, \quad (9)$$

where a_i is the weight of its respective local ML model. The local ML models may have different weights based on the aggregation scheme, but the sum of all the weights must equal one. w_i indicates a local ML model. w_g is the global ML model. From this perspective, all existing aggregation schemes such as FedAVG [17] are linear combinations. However, they do not reduce the linear combination to a simplified form so that the performance could be further optimised. We propose employing linear combination techniques to optimise the aggregation process to filter out malicious and redundant ML models.

According to Definition 4 and Theorem 1, there exists a subset of local ML models that is a basis of the vector space of the ML models, and they are linear independent. The linear combination of the vectors from the basis can represent other vectors in this vector space. In this solution, we find out independent linear combinations of local ML models and filter out those that are not when performing aggregation in FL. The intuition is that if we can carry out a global ML model by combining only necessary local ML models, then there is no need to include other local ML models that are redundant and may be malicious.

Since the set of linear independent local ML models (i.e., vectors) is a basis, the coefficients (i.e., aggregation weights of local ML models) are the coordinates of the dependent vectors. Figure 1 shows an example of this idea. The three vectors with colours of red, green and blue (i.e., Vector 1, 2, and 3) form a basis of the three-dimensional vector space. The grey vectors (i.e., Vector 4, 5, 6 and Malicious) are dependent vectors that can be represented by the linear combination

of the three base vectors (i.e., Vector 1, 2, and 3). The coordinates (i.e., the coefficients of the linear combination) in the vector space can uniquely identify a grey vector. One of the grey vectors is malicious, but we only consider independent vectors, so the malicious vector will be filtered out during the aggregation process.
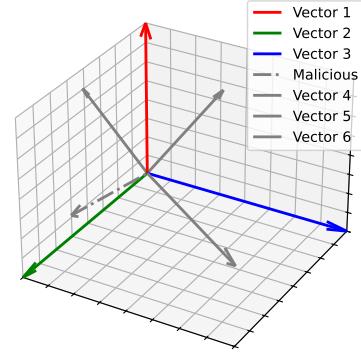

Figure 1. Vectors in a 3-Dimensional Vector Space.

If a dependent vector can be represented by a linear combination, then it means the information of the dependent vector is contained in the basis and coordinate system. Hence, the dependent vectors can be regarded as redundant vectors that may contain malicious data and should be filtered out. This idea is backed up by the row reduced echelon form (RREF) of a matrix since a matrix can be viewed as a set of vectors. The RREF of a matrix is a form that only keeps the pivot rows of the matrix while other rows are zeroed out [15, p. 30]. The pivot rows contain all the necessary information in a matrix to solve a linear algebra problem [15, p. 30].

3.3.2 Solution Overview

Based on the above foundation, in FedLAD, we organise local ML models into a matrix and compute its RREF. The list of pivot rows (also simplified as pivot) of the RREF is the independent linear combination of the local ML models. We only select local ML models corresponding to these pivot rows for the aggregation process in FL. Hence, the impact of malicious ML models caused by data poisoning attacks can be eliminated.

Figure 2 shows the overview of the process of FedLAD. There are three nodes: A, B, and C, with node C being malicious. The circled numbers indicate the step of the process. In the first step, the nodes submit their local ML models to a server. In the second step, the server flattens the ML models into vectors and lists them as a matrix. In the third step, the server finds the independent ML model vectors by calculating the RREF of the matrix and then selects only local ML models from the pivot rows in the RREF. In the last step, the selected local ML models are merged into a global ML model. The following sections cover technical details in the process.

3.3.3 Flatten ML models

ML models may have multiple layers, and we need to flatten them to one-dimensional (1-D) vectors. In FedLAD, we flatten ML models into 1-D vectors based on row-major (C-style) order [14, p. 253].

3.3.4 Calculate RREF

We apply Gaussian Elimination (GE) [15, p. 30] to calculate the RREF of a matrix. Algorithm 1 shows the detailed pseudo code
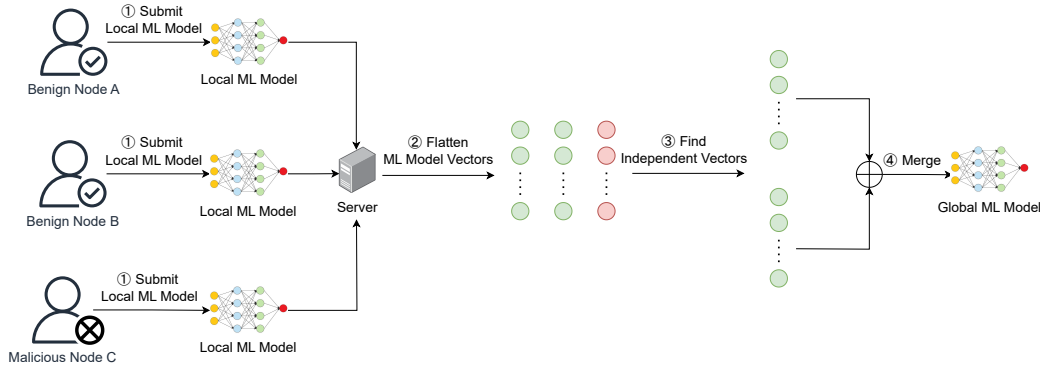
Figure 2. The overview of FedLAD.

of calculating RREF for a given matrix. It finds possible pivots by searching from each column and row. The "cross cancel" function in Algorithm 2 zeros out other elements except the pivot in a row to make sure only the pivot is non-zero based on the definition of RREF [15, p. 30]. According to [15, p. 233], the RREF for a matrix is unique, so there is no need to handle possible multiple forms of independent linear combinations.

This paragraph details how RREF is calculated in Algorithm 1. Lines 5 and 6 iterate through the columns and rows. Line 8 retrieves the rows of the current column that are below the currently processed row. Line 10 finds the index of the maximal number regardless of the number sign (i.e., positive or negative) from the rows obtained from line 8. Line 11 determines whether the index in line 10 is a possible pivot index. The number in the index must not be zero otherwise, its rank will be zero. Lines from 17 to 19 swap the row from the index in line 12 with the current row. Lines from 20 to 28 zero out the rows except the current row to ensure only the pivot row is non-zero, which is a characteristic of RREF [15, p. 30]. The steps from 17 to 19 and from 20 to 28 are the GE steps to eliminate unnecessary rows. In line 29, we continue to search for other possible pivots after the current row. In line 32, if no possible pivot is found, we perform an early stop to conserve computational resources and move on to the next column. After finishing the iteration of columns and rows, the found pivots are returned.

3.3.5 Global ML Model Aggregation

After the RREF of the matrix of local ML models is carried out, we apply FedAVG on the local ML models that are the RREF pivots. It means that the elements of the coordinate in the basis are the same and the scale is normalised to one. The rationale is that information from the independent vectors is considered equally important. By doing so, we can limit the influence of malicious ML models while ensuring that valuable information from normal ML models is not overlooked.

3.3.6 Computation Optimisation

As the size of a matrix increases, the time required for computation also increases. Algorithm 1 addresses this issue by employing an early stopping strategy, which allows the algorithm to break out early if it determines that a row is certainly not a pivot. This helps avoid unnecessary calculations. However, this algorithm serially iterates through the columns and rows without parallelism, resulting in wasted idle computing resources in modern parallel computing platforms. To improve efficiency and reduce computation time, we propose a parallel version to perform this task.

Algorithm 1 Calculate RREF

Input: Local ML Models $W = [w_1, w_2, ..., w_n]^T$
Output: RREF pivots

1: let pivot_cols $= [\,]$
2: let row_index $= 0$
3: let row_size be the number of rows of W
4: let col_size be the number of columns of W
5: **for** col in $\{0, ..., \text{col\_size}\}$ **do**
6:   **for** row in $\{\text{row\_index}, ..., \text{row\_size}\}$ **do**
7:     // get the rows of the current column in the iteration
8:     let row_col $= W[row :, col]$
9:     // find the possible pivot index whose value must not be zero
10:     let possible_pivot_index $= argmax(abs(\text{row\_col}))$
11:     **if** $abs(\text{row\_col}[\text{possible\_pivot\_index}]) > 0$ **then**
12:       let pivot_index $= $ possible_pivot_index
13:       // add the pivot to the list
14:       add col to pivot_cols
15:       // exchange the pivot row with the current row based on GE operations
16:       **if** pivot_index $\neq 0$ **then**
17:         let $t = W[row]$
18:         let $W[row] = W[row + \text{pivot\_index}]$
19:         let $W[row + \text{pivot\_index}] = t$
20:       let row_above $= W[: row]$
21:       let row_below $= W[row + 1 :]$
22:       let current_row_col_value $= W[row, col]$
23:       // zero out rows above the pivot
24:       **if** the row size of row_above > 0 **then**
25:         $W[:\ row] = $ cross_cancel(current_row_col_value, $W[row]$, row_above, col)
26:       // zero out rows below the pivot
27:       **if** the row size of row_below > 0 **then**
28:         $W[row+1 :] = $ cross_cancel(current_row_col_value, $W[row]$, row_below, col)
29:       row_index $+ = 1$
30:     **else**
31:       // early stop if no pivot found
32:       break
33: **return** pivot_cols

Algorithm 2 Cross Cancel

Input: Current Value v, Current Row row, Considered Rows rows, Current Column col
Output: Cross Cancelled Rows

1: repeat rows$[:, col]$ horizontally and turn them into a square matrix and set it as a
2: repeat row horizontally and turn them into a matrix with the column size as same as the row size of a and set it as b
3: let $c = a \times b$
4: **return** $v \times$ rows $- c$

Algorithm 3 outlines the details of this approach. It involves splitting the given matrix into sub-matrices, either by columns or by rows. The example in Algorithm 3 illustrates a column-wise split, but a row-wise split would yield the same result [6, p. 123]. Once the matrix is divided into sub-matrices, multiple computing units (CPUs) can process different sub-matrices simultaneously. After calculating the RREF for each sub-matrix, we combine these results into a new matrix and repeat the process of splitting and combining until we are left with the smallest-sized sub-matrix. The RREF from this smallest sub-matrix is equivalent to that of the original matrix. We provide a proof of this in the **supplementary document Section 1**. For Fed-LAD, the result of the parallel RREF calculation is the same as that of the non-parallel version since the same RREF form results in the same pivots according to the definition of RREF [15, p. 233].

Algorithm 3 Parallel RREF Calculation

Input: Local ML Models $W = [w_1, w_2, ..., w_n]^T$
Output: RREF pivots

1: let n be the number of CPUs
2: let col_size be the number of columns of W
3: let smallest_submatrix_size = col_size $\div n$
4: split W into n sub matrices based on the split size smallest_submatrix_size and assign them to sub_matrices[ ]
5: spawn n CPU threads to call Algorithm 1 and pass on the respective input of sub_matrices$[i]$ with $i \in n$
6: combine the returned RREFs from the threads to a new matrix W_s
7: **if** the size of $W_s >$ smallest_submatrix_size **then**
8:     call Algorithm 3 and pass on the input of W_s
9: **else**
10:     call Algorithm 1 and pass on the input of W_s
11: **return** the RREF returned form calling Algorithm 1

---

4 Experiment Evaluation

4.1 Experiment Settings

We conduct experiments with 10 nodes, which are initialised with an array of local ML models, datasets, and node ID, to evaluate our proposed method FedLAD. The ratio of malicious nodes ranges from 20% to 80% (we also use floating numbers 0.2 to 0.8 to indicate the ratio), which is to verify the robustness of the proposed defence method and baseline defence methods. The malicious attack performs label flipping on the experimental datasets. The details of the label flipping are introduced hereinafter. We demonstrate the improvements of FedLAD by comparing it with five baseline methods:

Sherpa [18], CONTRA [1], Median [24], Trimmed Mean [23], and Krum [5]. The justification for choosing these baseline methods is provided in the **supplementary document Section 4**. We conduct the experiments five times and adopt the average of the results. We disclose the standard deviation alongside the average of the experiment results in the **supplementary document Section 5**. We also disclose the running time of FedLAD and baselines in Section 4.2.3. We run FedLAD on serial and parallel versions and list the respective running times to demonstrate the advantage of parallel optimisation proposed in Section 3.3.6.

4.1.1 Computational and Software Setups

The experiments are carried out on a high-performance computer with 64GB of memory, 16 CPUs and an A40 GPU to accelerate multiple-node federated learning with large-scale neural networks. The implementations of FedLAD and baseline methods are based on Pytorch 2.3.1, Numpy 1.26.3, Flower 1.15.0 (a federated learning framework) and Python 3.10.12.

4.1.2 Implementation Details

To evaluate FedLAD in various scenarios, we use two image datasets, CIFAR10 and CIFAR100, [1] and an NLP dataset, AG_NEWS. [2] To enable non-IID distribution in the datasets, we apply a Dirichlet distribution [11, 7] with the heterogeneity parameter set to 0.5 to sample different data distributions for each node. Following the work [1], we use ResNet18 for the image recognition tasks on the CIFAR10 and CIFAR100 datasets. For the AG_NEWS dataset, we use a three-layer neural network, which is the optimal structure according to our experiment, to perform the text classification task. The input layer of this neural network has 95,811 units, corresponding to the number of tokens in the AG_NEWS dataset. The hidden layer contains 64 units, which embeds the input text into a vector space, and the output layer has four units, corresponding to the four classes. The work [1] used 300 rounds for global communication and one round for local training. We adopt the same strategy but with some adaptations based on our experimental observation, in which the ML model performance is optimal. For the CIFAR10 and CIFAR100 datasets, we trained for two rounds in the local environment and conducted 40 rounds of global communication. For the AG_NEWS dataset, we trained for two local rounds and conducted 20 rounds of global communication. During the training, malicious nodes perform label-flipping attacks on their local dataset.

4.1.3 Malicious Nodes Training

The details of how malicious nodes are generated are in the **supplementary document Section 2**. When malicious nodes train their local machine learning (ML) models, they perform label flipping on the training data. Following [18], we swap two targeted labels of the training dataset.

4.1.4 Performance Metrics

To evaluate the performance of FedLAD to defend against data poisoning attacks, we adopt two metrics: attack success rate (ASR) and model accuracy (MA). The details, such as their definitions of these metrics, are in the **supplementary document Section 3**.

---

[1] https://www.cs.toronto.edu/˜kriz/cifar.html
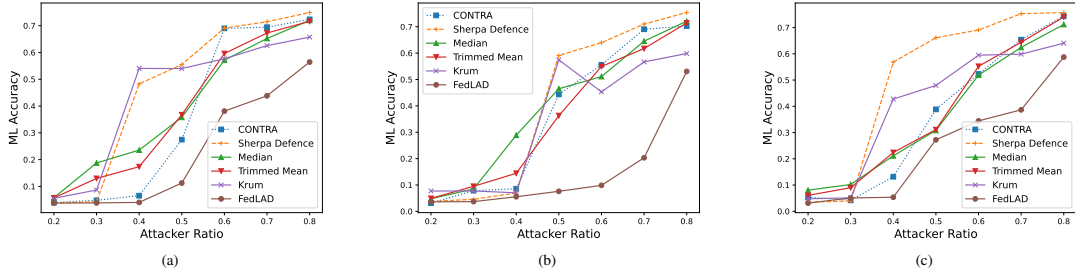[2] http://groups.di.unipi.it/˜gulli/AG_corpus_of_news_articles.html

Figure 3. Malicious ratio VS Attacking Success Rate on Non-IID datasets. 3a is under CIFAR10. 3b is under CIFAR100. 3c is under AG_NEWS
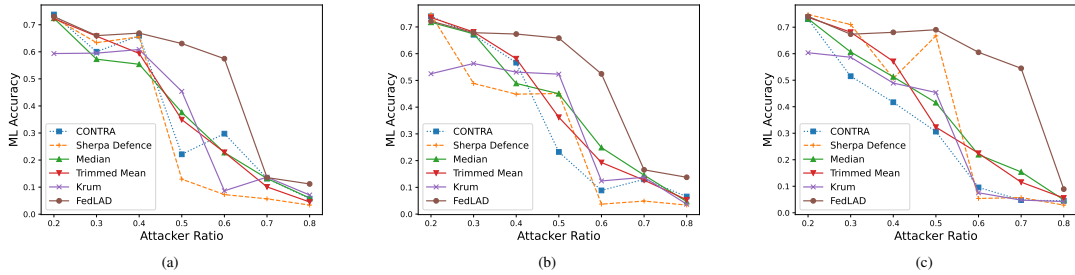

Figure 4. Malicious ratio VS Model Accuracy on Non-IID datasets. 4a is under CIFAR10. 4b is under CIFAR100. 4c is under AG_NEWS

4.2 Experiment Results and Evaluation

4.2.1 Attacking Success Rate (ASR)

Figure 3 shows the ASR for malicious ratios ranging from 0.2 to 0.8 for the CIFAR10, CIFAR100, and AG_NEWS datasets. Compared with baseline methods, our method, FedLAD, exhibits the highest performance in resisting malicious nodes across various ratio settings, showcasing the advantage of resisting attacks from malicious nodes. The ASR of FedLAD increases sharply when the malicious ratio exceeds 0.5 on CIFAR10 and AG_NEWS, 0.7 on CIFAR100, as there are too few benign ML models to cancel out the attacking effect of malicious ML models. The reason why FedLAD maintains a lower ASR until the malicious ratio surpasses 0.7 is that the CIFAR100 dataset has 100 labels, which limits the impact of label flipping attacks from malicious nodes.

All baseline methods show a high ASR when the malicious ratio exceeds 0.5 (i.e., ASR increases sharply). This is because these methods rely on the majority to be benign nodes, which fail when the malicious ratio surpasses 0.5. Among the baseline methods, Sherpa performs the worst probability due to that the calculation of SHAP features is based on sampling methods, which can result in some important features being omitted. Krum performs reasonably better than other baseline methods probability due to the effectiveness of excluding thirty per cent of nodes (i.e., a Byzantine tolerant setting in Krum). However, the MA performance of Krum is poorer than other baseline methods as we can see in Section 4.2.2.

4.2.2 Model Accuracy (MA)

Figure 4 shows the MA with malicious ratios ranging from 0.2 to 0.8 for the CIFAR10, CIFAR100, and AG_NEWS datasets. Compared with baseline methods, FedLAD maintains the highest performance across all malicious ratio settings, demonstrating that FedLAD can lower the impact of malicious attacks on the MA. Performance of FedLAD drops sharply when the malicious ratio is 0.7 and 0.8. This is because there are not enough benign nodes to contribute to the MA performance. Despite this, FedLAD can still maintain high performance even when the ratio is as high as 0.8, as only the pivot ML models are selected, which can exclude redundant and malicious ML models, even at high malicious ratios.

Among the baseline models, Krum performs the worst as it maintains a low MA across the three datasets probably due to the thirty per cent of nodes are excluded. Some of these excluded nodes may contain the most important contribution to MA. FedLAD can avoid this situation since the most important contributions are always included in the RREF pivots according to the mechanism of independent linear combination in linear algebra. Sherpa slightly outperforms other baseline methods under the AG_NEWS dataset, but with a high variance. The reason may be that the texts are encoded into vectors by an encoder. Some important information is retained, and the noise is ignored. Compared with the pixel features in pictures, the vector features are more efficient as an input to produce the SHAP [13] features.

4.2.3 Running Time of Defence Methods

Table 1 presents the running times of the defence methods on the CIFAR10, CIFAR100, and AG_NEWS datasets. The running time is recorded during the five runs of the experiments, and we adopt the average value. The serial version of FedLAD has the highest running time. However, when parallel optimisation is applied, there is a significant reduction in running time, demonstrating the effectiveness of this approach. Even though the running time of the parallel optimised FedLAD does not surpass that of the baseline methods due to the complexity of linear algebra operations and communication between parallel CPU threads, the overall time consumption is acceptable considering that FedLAD offers a more robust defence against malicious attacks in Federated Learning compared to the baseline methods.

5 Related Works

We classify defence methods into two categories based on the underlying technical approach: clustering-alike defence and robust training-based defence.

Table 1. Running Time (Minutes) of Defence Methods on Three datasets

Method	CIFAR10	CIFAR100	AG_NEWS
FedLAD Serial	**245.63**	**281.44**	**127.51**
FedLAD Parallel	**63.43**	**73.88**	**36.7**
Sherpa [18]	58.05	63.45	35.97
CONTRA [1]	49.01	67.62	25.72
Median [24]	44.14	47.57	27.48
Trimmed Mean [23]	44.34	72.05	29.43
Krum [5]	45.51	68.72	26.62

5.1 Clustering Alike Defence

This category reviews works that determine whether a node is malicious based on its relevance to the majority, using techniques such as cosine similarity and machine learning model interpretability to defend against malicious nodes, similar to clustering. Tolpegin et al. [20] investigate how malicious participants can compromise federated learning by introducing poisoned data. Their study focuses on label-flipping attacks, where attackers modify class labels in their local datasets to manipulate the global model. They propose a defence mechanism using principal component analysis (PCA) to detect anomalous updates from malicious participants, demonstrating its effectiveness in isolating such participants. MOAT [16] analyses the interpretability of machine learning models based on SHAP (i.e., Shapley Additive Explanations) features [13]. It posits that malicious and benign models have different interpretabilities, as SHAP features reveal these differences. However, MOAT relies on Z-scores and a threshold to determine whether a model is malicious, which may not be consistent across all datasets, as pointed out by SHERPA [18]. To address this issue, SHERPA improves MOAT by incorporating the HDBSCAN clustering method and a scoring mechanism that is less sensitive to the variability of different datasets. CONTRA [1] employs a cosine-similarity-based measure to assess the credibility of local model parameters, alongside a reputation scheme to dynamically adjust the contributions of each node. This method assumes that benign nodes will be more similar to the majority compared to malicious nodes. Extensive experiments show that CONTRA significantly reduces the attack success rate and minimises degradation of global model performance compared to state-of-the-art defences. MCDFL [8], which stands for Malicious Clients Detection Federated Learning, proposes a defence mechanism against label-flipping attacks in federated learning. By recovering a distribution over a latent feature space, MCDFL can identify malicious nodes and assess the data quality of each node. It assumes that benign nodes will exhibit similar data quality to the majority, while malicious nodes will not. The proposed strategy is tested on benchmark datasets such as CIFAR10 and Fashion MNIST, using different neural network models and attack scenarios. Results show that this solution is robust in detecting malicious nodes without incurring excessive costs. VFedAD [9], meaning a Defence Method Based on the Information Mechanism Behind Vertical Federated Data Poisoning Attacks, addresses the challenges posed by vertical federated data poisoning attacks. It proposes an unsupervised defence method rooted in information theory that learns semantic-rich representations of client data to effectively detect anomalies, thereby protecting subsequent algorithms from attacks. Similar to MCDFL, it assumes that benign nodes will have data representations similar to the majority, while malicious nodes will not. Experimental results validate VFedAD's ability to detect anomalies introduced by data poisoning attacks, demonstrating its effectiveness as a defence mechanism. However, these methods may struggle in scenarios with a high ratio of malicious nodes, as a significant percentage of malicious nodes can dominate and distort the standard used for comparison among nodes.

5.2 Robust Training Based Defence

This category focuses on studies that utilise machine learning model aggregation strategies and noise injection techniques to mitigate the impact of malicious nodes. Krum [5] proposed an aggregation rule (i.e., a method for merging local machine learning models) that first calculates the squared sum of distances between the weight vector of the current node's model and the weight vectors of the $n-f-2$ closest nodes. The rule then selects the weight vector with the smallest squared sum distance as the global model's weight vector. The work by Yin et al. [24] introduced a median-based merging method. This approach uses the coordinate-wise median value to replace the average value used in [17] when merging local models into a global model. Trimmed Mean [23] proposes a method for trimming the model weight values based on a predefined trim parameter K, with the condition that $k < \frac{n}{2}$, where n is the number of nodes. The selected model weights, determined by the trim parameter, are then aggregated into a global model using a mean aggregation scheme. FRIENDS [12] is a noise injection defence mechanism, short for Friendly Noise Defence, designed to counter data poisoning attacks in deep learning. The authors identify that poisoning attacks create local sharp regions with high training loss, which facilitate adversarial perturbations. To neutralise this, FRIENDS employs a two-part noise strategy: (1) Friendly noise, which introduces optimised perturbations that do not degrade model performance but disrupt attack success, and (2) Random noise, which varies across training iterations to prevent adaptive attacks from overcoming the defence. FRIENDS effectively defends against major poisoning attacks while maintaining high model accuracy and minimal computational overhead. It is also transferable across different architectures, making it practical for real-world deep learning applications. However, the introduction of noise adds additional complexities during training and inevitably affects the model's accuracy.

Existing methods for detecting malicious nodes have a significant limitation: their effectiveness declines when the proportion of malicious nodes exceeds a certain threshold. For instance, clustering-alike defence methods become ineffective when more than 50% of the group consists of malicious nodes, as these methods primarily focus on the majority class. When the majority of nodes are malicious, the algorithms struggle to accurately classify which nodes are benign and which are malicious. Similarly, robust training methods can be overwhelmed when the majority of nodes are malicious. Additionally, these methods are typically based on serial processing and do not support parallel optimisation, resulting in low computational efficiency.

6 Conclusion

This paper described the FedLAD data poisoning defence method for FL. FedLAD exhibits high tolerance to malicious nodes in terms of data poisoning in an FL system, thereby providing greater robustness. Comprehensive experiments demonstrated the advantages of our method with various malicious settings. While this paper applies FedAVG on the RREF pivots, our future work will explore other aggregation schemes such as Median, Krum, etc. to study further possible improvements of the robustness of FedLAD.

References

[1] S. Awan, B. Luo, and F. Li. CONTRA: Defending Against Poisoning Attacks in Federated Learning. In *Computer Security – ESORICS 2021*, pages 455–475, 2021.

[2] S. Axler. *Linear algebra done right*. Springer Nature, 2024.

[3] E. Bagdasaryan and V. Shmatikov. Blind backdoors in deep learning models. In *USENIX Security Symposium (USENIX-Security)*, pages 1505–1521, 2021.

[4] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. In *Intl. Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2938–2948, 2020.

[5] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, 2017.

[6] J. Gentle. *Matrix Algebra: Theory, Computations and Applications in Statistics*. Springer Texts in Statistics. Springer International Publishing, 2024. ISBN 9783031421440. URL https://books.google.com.au/books?id=HF38EAAAQBAJ.

[7] S. Han, S. Park, F. Wu, S. Kim, C. Wu, X. Xie, and M. Cha. Fedx: Unsupervised federated learning with cross knowledge distillation. In *European Conference on Computer Vision (ECCV)*, pages 691–707, 2022.

[8] Y. Jiang, W. Zhang, and Y. Chen. Data Quality Detection Mechanism Against Label Flipping Attacks in Federated Learning. *IEEE Transactions on Information Forensics and Security*, 18:1625–1637, 2023.

[9] J. Lai, T. Wang, C. Chen, Y. Li, and Z. Zheng. VFedAD: A Defense Method Based on the Information Mechanism Behind the Vertical Federated Data Poisoning Attack. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1148–1157, 2023.

[10] B. N. Levine, C. Shields, and N. B. Margolin. A survey of solutions to the sybil attack. *University of Massachusetts Amherst, Amherst, MA*, 7: 224, 2006.

[11] Q. Li, B. He, and D. Song. Model-contrastive federated learning. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10713–10722, 2021.

[12] T. Y. Liu, Y. Yang, and B. Mirzasoleiman. Friendly Noise against Adversarial Noise: A Powerful Defense against Data Poisoning Attack. *Advances in Neural Information Processing Systems*, 35:11947–11959, 2022.

[13] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[14] R. Manglik, E. Experts, and E. Community. *Introduction to Algorithms*. EduGorilla Publication, 2024. ISBN 9789358800043. URL https://books.google.com.au/books?id=zgU8EQAAQBAJ.

[15] R. Manglik, E. Experts, and E. Community. *Linear Algebra and Matrix theory*. EduGorilla Publication, 2024. ISBN 9789366896090. URL https://books.google.com.au/books?id=hFE6EQAAQBAJ.

[16] A. Manna, H. Kasyap, and S. Tripathy. Moat: Model agnostic defense against targeted poisoning attacks in federated learning. In *Information and Communications Security: 23rd International Conference, ICICS 2021, Chongqing, China, November 19-21, 2021, Proceedings, Part I 23*, pages 38–55. Springer, 2021.

[17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Intl. Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282, 2017.

[18] C. Sandeepa, B. Siniarski, S. Wang, and M. Liyanage. SHERPA: Explainable Robust Algorithms for Privacy-Preserved Federated Learning in Future Networks to Defend Against Data Poisoning Attacks. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 4772–4790, 2024.

[19] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu. Data poisoning attacks against federated learning systems. In *Proc. of the European Symposium on Research in Computer Security (ESORICS)*, pages 480–501, 2020.

[20] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu. Data Poisoning Attacks Against Federated Learning Systems. In *Computer Security – ESORICS 2020*, pages 480–501, 2020.

[21] X. Xiao, Z. Tang, C. Li, B. Xiao, and K. Li. Sca: Sybil-based collusion attacks of iiot data poisoning in federated learning. *IEEE Transactions on Industrial Informatics*, 19(3):2608–2618, 2023.

[22] C. Xie, O. Koyejo, and I. Gupta. Generalized Byzantine-tolerant SGD. *arXiv preprint arXiv:1802.10116*, 2018.

[23] D. Yin, Y. Chen, R. Kannan, and P. Bartlett. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5650–5659, 2018.

[24] D. Yin, Y. Chen, R. Kannan, and P. Bartlett. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *Proc. of the Intl. Conference on Machine Learning (ICML)*, pages 5650–5659, 2018.